TREMULOUS 1.1.0

March 17, 2006

Contents

1 Introduction

Tremulous is a first person shooter featuring two opposing teams, humans and aliens. Both teams are able to build structures such as spawn points, which are vital to their victory. The goal of Tremulous is to eliminate the opposing team and all of their spawn points.

Each team in Tremulous differs fundamentally from the other. The aliens are class based, with two classes initially available: the alien team's builder, known as the Granger, and the Dretch, the weakest offensive alien. The aliens are awarded frags for killing their foes which may be used to evolve into stronger classes, capable of greater and more varied maneuvers. In contrast the human team is upgrade based, receiving credits for kills that may be exchanged at an Armoury structure for new weapons, armour and equipment. Two such upgrades are available for free: a rifle and a construction kit, used for building structures.

During a game of Tremulous, each team occupies one of three stages of development. These stages are reached by accruing more than a specific total number of kills by the whole team. Each new stage unlocks new classes, upgrades and buildable structures. If one team reaches a stage significantly earlier than the other team it stands a better chance of defeating the opposing team.

Section 2 details the content of the game including the various controls that are used to play. Section 3 describes some technical aspects of how Tremulous works. It is not necessary to read this section in order to play the game.

2 Game

2.1 Aliens

Two classes are available upon joining the alien team: the Dretch and the Granger. As you gain kills, you may use your earned frags to evolve into higher classes with the USE STRUCTURE/EVOLVE button. The alien team is mostly limited to melee attacks and must use stealth and speed to defeat the longer range humans. All aliens automatically regenerate health at a slow rate.

2.1.1 Classes

Granger

Cost: 0 Stage: 1

Ability	Control
Build	PRIMARY ATTACK
Destroy Structure	DECONSTRUCT STRUCTURE on an alien structure

The *Granger* is the alien team's builder class. PRIMARY ATTACK will bring up a menu of structures available for building. After selecting a structure a glowing outline of it will appear. When this outline is green you can use the PRIMARY ATTACK button to place it. The outline changes to red when the structure cannot be placed in its current location. To cancel placing the structure press the SECONDARY ATTACK button. To remove a placed structure use the DECONSTRUCT STRUCTURE button. After building or deconstructing a structure a timer will appear in the lower right corner of the screen. Until this timer expires you cannot create or destroy another building.

Advanced Granger

Cost: 0 Stage: 2

Ability	Control
Build	PRIMARY ATTACK
Slash	SECONDARY ATTACK
Lob Projectile	ACTIVATE UPGRADE
Destroy Structure	DECONSTRUCT STRUCTURE on an alien structure
Wallwalk	CROUCH

The *Advanced Granger* becomes available at no cost when the alien team reaches stage two. In addition to the *Granger's* abilities, the *Advanced Granger* can move faster, jump higher, walk on walls and attack with a slash or by lobbing small projectiles with the ACTIVATE UPGRADE button.

Dretch

Cost: 0 Stage: 1

Ability	Control
Bite	Touch a human
Wallwalk	CROUCH

The *Dretch* is the alien team's weakest offensive class. Its only attack is to make forward contact with a human player or human defensive structure. The amount of damage dealt to a human depends on what armour they were wearing and where they were hit, with headshots resulting in the most damage. *Dretches* can also wallwalk; toggle it by pressing the CROUCH button.

Basilisk

Cost: 1 Stage: 1

Ability	Control
Slash	PRIMARY ATTACK
Grab	Touch a human
Wallwalk	CROUCH

The *Basilisk* attacks by using the PRIMARY ATTACK button. It can also grab human players by making contact with them at close range. This freezes humans in place and, if they're not wearing a *Battlesuit*, restricts their ability to turn. The *Basilisk* can also wallwalk; toggle it by pressing the CROUCH button.

Advanced Basilisk

Cost: 2 Stage: 2

Ability	Control
Slash	PRIMARY ATTACK
Gas	SECONDARY ATTACK
Grab	Touch a human
Wallwalk	CROUCH

In addition to the *Basilisk's* abilities, the *Advanced Basilisk* can spray a cloud of noxious gas that will disorient and poison affected human players. Humans equipped with a *Battlesuit* are immune to gas. Use this ability with the SECONDARY ATTACK button.

Marauder

Cost: 2 Stage: 1

Ability	Control
Bite	PRIMARY ATTACK
Wall Jump	Jump into a wall while holding down JUMP

The *Marauder* attacks with the PRIMARY ATTACK button and has the ability to rebound off walls. To use this ability jump towards a wall and hold down the JUMP button. When you hit the wall you will be propelled upward and in the direction opposite of the wall. As long as you continue hitting walls you will continue wall jumping.

Advanced Marauder

Cost: 3 Stage: 2

Ability	Control
Bite	PRIMARY ATTACK
Zap	SECONDARY ATTACK
Wall Jump	Jump into a wall while holding down JUMP

In addition to the *Marauder's* abilities, the *Advanced Marauder* can use a chain lightning attack. To use this, press the SECONDARY ATTACK button while aiming at a nearby human or human structure. If it connects the electric shock will jump to up to two other nearby targets doing full damage to the first, half damage to the second and one third damage to the third over a period of one second provided the attacker stays within range.

Dragoon

Cost: 3 Stage: 1

Ability	Control
Bite	PRIMARY ATTACK
Pounce	Hold down SECONDARY ATTACK briefly then release

The *Dragoon* attacks by either biting with the PRIMARY ATTACK button or pouncing with SECONDARY ATTACK. To pounce, first hold down the SECONDARY ATTACK button to charge up, then release to leap forward and damage anything that gets in the way. While charging you will be unable to jump normally and will move at a reduced rate. Aim up a little to fly further when pouncing.

Advanced Dragoon

Cost: 4 Stage: 3

Ability	Control
Bite	PRIMARY ATTACK
Pounce	Hold down SECONDARY ATTACK briefly then release
Shoot Barb	ACTIVATE UPGRADE

In addition to the *Dragoon's* abilities, the *Advanced Dragoon* can fire long ranged spiked barbs with the ACTIVATE UPGRADE button. Up to three of these barbs may be held in reserve and they regenerate automatically over time.

Tyrant

Cost: 5 Stage: 3

Ability	Control
Slash	PRIMARY ATTACK
Trample	Hold down SECONDARY ATTACK briefly then release
Healing Aura	Stand close to teammates to increase their regeneration rate

The *Tyrant* attacks by either slashing with the PRIMARY ATTACK button or trampling with SECONDARY ATTACK. To trample, first hold down the SECONDARY ATTACK button while moving forward to charge up, then release to run at high speed for a short time, damaging anything in your path. The *Tyrant* also has a healing aura that will double the regeneration rate of lower class aliens within range.

2.1.2 Structures

All alien structures must be built in proximity to an *Egg* or an *Overmind*. All alien structures require the presence of an *Overmind* to function. All alien structures create 'creep' around their bases that slows human movement. When destroyed, alien structures explode in a shower of acid harmful to humans. All structures may be built on level floors and when *Advanced Grangers* become available some structures may also be built on walls and ceilings.

Overmind

Sentience: 0 Stage: 1

The *Overmind* is the collective consciousness that controls all the alien structures in a map and enables aliens to evolve into higher forms. There can only be one *Overmind* and it must be alive before any other structures can be built. If the *Overmind* is destroyed then all structures besides *Eggs* cease to function and every alien loses the ability to upgrade their class until a new *Overmind* is built. The *Overmind* has a limited amount of 'sentience' which is distributed amongst every other structure built, each having its own cost.

Egg

Sentience: 10 Stage: 1

The *Egg* is the most basic and important alien structure; it is from these that aliens spawn into the game. They are also the only structure that continues to function in the absence of an *Overmind*. *Eggs* may be built on ceilings.

Acid Tube

Sentience: 8 Stage: 1

Acid Tubes are the primary defensive structure for the alien team. When approached by a human they eject lethal acid in all directions, even over other structures. *Acid Tubes* may be built on walls and ceilings.

Barricade

Sentience: 10 Stage: 1

Barricades are used to obstruct corridors and doorways, hindering human movement and line-of-sight.

Trapper

Sentience: 8 Stage: 2

Trappers fire a blob of adhesive spit at any human in their line of sight, freezing them in place and, if they're not wearing a *Battlesuit*, restricts their ability to turn. *Trappers* may be built on walls and ceilings, and are rarely effective when built on floors.

Booster

Sentience: 12 Stage: 2

Any alien that touches a *Booster* is provided with a poison enhancement on all their melee attacks for a limited time. Poison causes victims to lose health steadily over time unless they use a *Medkit* or visit a *Medistation*. Poison does not work against humans equipped with a *Battlesuit*. The *Booster* will also double the regeneration rate of any nearby aliens with the exception of *Tyrants*. The healing aura of a *Tyrant* is not cumulative with the healing effect of boosters.

Hovel

Sentience: 0 Stage: 3

Hovels are armored shells that *Grangers* may hide in should the need arise. There can only be one Hovel. They may be entered and exited with the USE STRUCTURE/EVOLVE button.

Hive

Sentience: 12 Stage: 3

Hives house millions of tiny insectoid aliens. When a human approaches the structure the insects attack. *Hives* may be built on ceilings.

2.2 Humans

2.2.1 Weapons

Humans may spawn with either the *Construction Kit* or the *Rifle*. As credits are earned, humans may sell their old upgrades and purchase new ones at an *Armoury* structure. Ammo may be refilled for normal weapons at *Armouries*, or at *Reactors* and *Repeaters* for energy weapons, all at no cost. Players may only carry one weapon at a time, excluding the *Blaster*. In general the humans rely on long range weapons to make up for their lack of mobility relative to the alien team.

Construction Kit

Cost: 0 Stage: 1

The *Construction Kit* is the humans' method of building structures. The PRIMARY ATTACK button will bring up a menu of structures available for building. After selecting a structure, a glowing outline of it will appear. When this outline is

green, pressing the PRIMARY ATTACK button will place it. When the outline is red, the structure cannot be placed in its current location. To cancel placing the structure, press the SECONDARY ATTACK button. To remove a placed structure, use the DECONSTRUCT STRUCTURE button. After building or deconstructing a structure, a timer will show up in the lower right corner of the screen. Until this timer expires, you cannot create, destroy or repair any structures. Damaged structures may otherwise be repaired with the SECONDARY ATTACK button.

Advanced Construction Kit

Cost: 0 Stage: 2

At stage two an upgraded *Construction Kit* becomes available that allows the building of more advanced structures.

Blaster

Cost: 0 Stage: 1

The *Blaster* is the human team's standard issue backup weapon. All players spawn with one automatically and may not exchange it for another weapon. The *Blaster* fires a weak projectile and uses no ammo.

Rifle

Cost: 0 Stage: 1

The *Rifle* is the human team's most basic weapon and is available from spawning. It rapidly fires moderately accurate shots with clip sizes of 30. Up to 6 extra clips may be carried at a time.

Pain Saw

Cost: 100 Stage: 1

The *Pain Saw* is a powerful melee weapon that emits a steady electric hum when in use. It uses no ammunition.

Shotgun

Cost: 150 Stage: 1

The *Shotgun* fires 8 pellets at a wide angle and is thus best used in close quarters. It holds 8 shots per clip and 3 extra clips may be carried at a time.

Las Gun

Cost: 250 Stage: 1

The *Las Gun* is similar to the *Rifle* but is more powerful, accurate, slower to fire and uses no clips. It is an energy weapon and so must be refilled at a *Reactor* or *Repeater*. It can hold up to 200 cells at a time, or 300 with a *Battery Pack*.

Mass Driver

Cost: 350 Stage: 1

The *Mass Driver* fires powerful, accurate shots at a slow rate of fire. It is an energy weapon and holds 5 shots per clip, or 7 with a *Battery Pack*. Up to 4 extra clips may be carried at a time.

Chaingun

Cost: 400 Stage: 1

The *Chaingun* is a powerful, wildly inaccurate rapid-fire weapon. It holds up to 300 bullets at a time and is best used when crouching to reduce its kickback. Humans equipped with a *Battlesuit* do not experience this kickback.

Pulse Rifle

Cost: 400 Stage: 2

The *Pulse Rifle* is an energy weapon that fires projectiles at high speeds. It holds up to 50 cells per clip, or 75 with a *Battery Pack*. Up to 4 extra clips may be carried at a time.

Grenade

Cost: 200 Stage: 2

The *Grenade* is a hand held explosive device. It is thrown for a short distance by using the ACTIVATE UPGRADE button. After a brief delay it will explode and cause tremendous damage to anything in its area of effect.

Flamethrower

Cost: 450 Stage: 3

The *Flamethrower* is a short range incendiary weapon. It holds up to 150 shots at a time and can easily damage the careless wielder.

Lucifer Cannon

Cost: 600 Stage: 3

The *Lucifer Cannon* is the human team's most devastating weapon. It is an energy weapon that can hold up to 90 cells at a time, or 135 with a *Battery Pack*. By holding down the PRIMARY ATTACK button, a player may charge up a powerful, slow moving projectile with splash damage. The longer the attack is charged the more powerful the projectile and the more ammo used. If the attack is charged for too long the weapon will explode, damaging the player. The SECONDARY ATTACK button fires a smaller projectile that requires no charging.

2.2.2 Upgrades

Human players may equip themselves with any number of the following upgrades, with a few exceptions: the *Jet Pack* and *Battery Pack* may not be used together and the *Battlesuit* may not be used with the *Jet Pack*, *Battery Pack*, *Light Armour*, or *Helmet*. Only one of any type of upgrade may be carried at a time. Upgrades that do not grant an intrinsic effect must be selected in the player's inventory with the NEXT UPGRADE and PREVIOUS UPGRADE buttons and then activated with the ACTIVATE UPGRADE button.

Light Armour

Cost: 70 Stage: 1

Light Armour grants the wearer improved defense to the torso and leg areas.

Helmet

Cost: 90 Stage: 2

The *Helmet* improves the defense of the wearer's head and also displays a radar that shows the relative positions of nearby enemies and enemy structures.

Medkit
Cost: 0 Stage: 1

Medkits are a free holdable given to every human upon spawning and refilled at *Medistations* to players with full health. They may not be refilled or exchanged at *Armouries*. When used with the ACTIVATE UPGRADE button, *Medkits* immediately begin restoring health at a slow rate, gradually speeding up until all damage incurred before the *Medkit* was activated is healed. Additionally, if a human is poisoned, using a *Medkit* will cure the poison and confer a 30 second immunity to poison.

Battery Pack

Cost: 100 Stage: 1

The *Battery Pack* increases the maximum ammo capacity of energy weapons by 50%. It may not be used in conjunction with the *Jet Pack*.

Jet Pack

Cost: 120 Stage: 2

The *Jet Pack* grants the wearer the power of slow but unlimited flight. When activated with the ACTIVATE UPGRADE button, a player may ascend or descend using the JUMP or CROUCH buttons, respectively. The *Jet Pack* ceases to function if there is no operational *Reactor* present. Additionally it temporarily cuts out if the player receives any damage. The *Jet Pack* may not be used in conjunction with the *Battery Pack*.

Battlesuit

Cost: 400 Stage: 3

The *Battlesuit* provides a significant defensive boost to the wearer's entire body. Due to this coverage, the *Battlesuit* may not be used in conjunction with any other wearable upgrade (*Light Armour*, *Helmet*, *Battery Pack*, and *Jet Pack*). Players are also prevented from crouching while wearing Battlesuits.

2.2.3 Structures

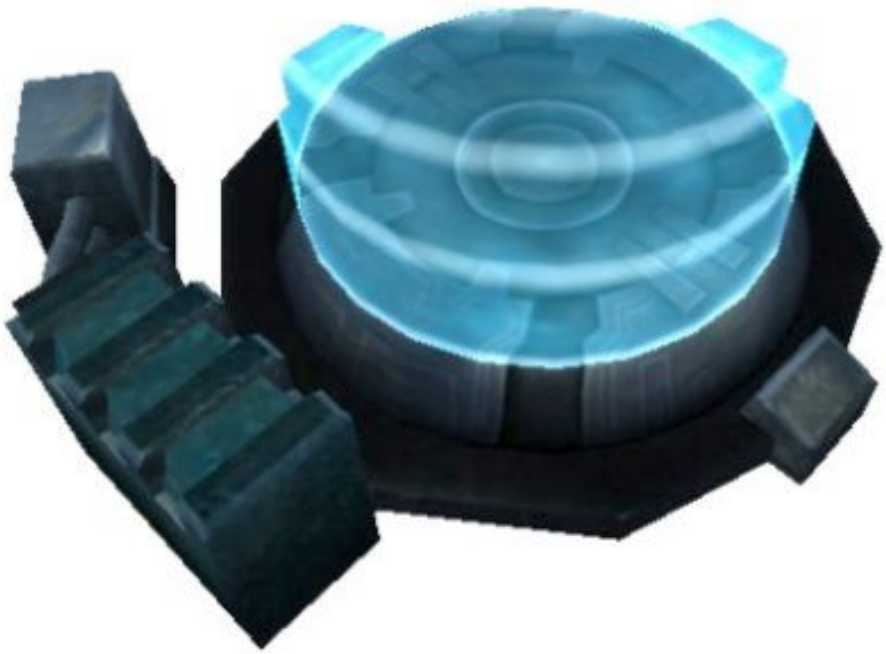
All human structures must be built in proximity to a *Reactor* or a *Repeater*. With *Telenodes* as the only exception, all structures require the presence of a working *Reactor* to function. All human structures explode in a powerful blast harmful to anything within their radius when destroyed.

Reactor

Power: 0 Stage: 1

The *Reactor* is the power source for all human structures in a map. There may only be one *Reactor*, and it must be present before any structures other than *Repeaters* can be built. If the *Reactor* is destroyed then all structures besides *Telenodes* cease to function. The *Reactor* has a limited amount of power which is distributed among every other structure built, each having its own cost.

Telenode



Power: 10 Stage: 1

The *Telenode* is the most basic and fundamental human structure; it is from these that humans spawn into the game. They are also the only structure that continues to function in the absence of a *Reactor*.

Machine Gun Turret

Power: 8 Stage: 1

The *Machine Gun Turret* is the primary defensive structure for the human team. While they have a clear line of sight to an alien within their range, they will track and fire at the alien until it is dead.

Tesla Generator

Power: 10 Stage: 3

Tesla Generators are a defensive structure that will unconditionally hit any target within their range and line of sight with an electrical surge. To be built and function, a *Tesla Generator* requires the presence of a *Defense Computer* somewhere in the map.

Armoury

Power: 10 Stage: 1

The *Armoury* is an essential part of every human base, allowing upgrades beyond the basic spawning equipment to be bought and exchanged. It is the sole means of human advancement. To use an *Armoury*, approach it and press the USE STRUCTURE/EVOLVE button. Ammo for non-energy weapons may also be acquired at no cost from an *Armoury* by using the BUY AMMO button.

Defense Computer

Power: 8 Stage: 2

Defense Computers coordinate the attacks of *Machine Gun Turrets*, preventing them from firing at a single target when multiple targets are available. They are also required for the production of *Tesla Generators*.

Medistation

Power: 8 Stage: 1

The *Medistation* provides the only means for humans to heal themselves. By standing on one, a human will quickly regenerate health up to their maximum of 100. *Medistations* will also refill *Medkits* to humans with full health. Only one person may use a *Medistation* at a time.

Repeater

Power: 0 Stage: 2

Repeaters serve as power distributors that may be built anywhere not already powered, even when no *Reactor* is present. Any other structure may be built in proximity to a working *Repeater* as if it were a *Reactor*. If a *Repeater* powers nothing for 90 seconds, it will automatically self destruct.

3 Technical

3.1 Bindings

Name in menu	Binding	Function
PRIMARY ATTACK	**+attack**	Use primary attack function.
SECONDARY ATTACK	**+button5**	Use secondary attack function.
PREVIOUS UPGRADE	**weapprev**	As human, preselect the previous upgrade in your inventory.
NEXT UPGRADE	**weapnext**	As human, preselect the next upgrade in your inventory.
ACTIVATE UPGRADE	**+button2**	As human, activate the current preselected inventory item. Also used for some alien abilities.
RELOAD	**reload**	As human, reload the selected weapon.
BUY AMMO	**buy ammo**	As human, buy ammo from an *armoury*, *repeater* or *reactor*.
USE MEDKIT	**itemact medkit**	As human, activate your *Medkit*.
USE STRUCTURE/EVOLVE	**+button7**	As human, use the structure in front of the player. As alien, evolve into a different class.
DECONSTRUCT STRUCTURE	**deconstruct**	As a builder class, deconstruct the structure in front of the player cleanly.
SPRINT	**boost**	Run faster.
–	**destroy**	As a builder class, destroy the structure in front of the player.
–	**itemact <item>**	If held, activate the specified item. For weapons this will select them.
–	**itemdeact <item>**	If held, deactivate the specified item.

–	**itemtoggle <item>**	If held, toggle the state of the specified item.
–	**sell <item>**	If held and within range of an *armoury*, sell the specified item.
–	**sell weapons**	If within range of an *armoury*, sell all weapons.
–	**sell upgrades**	If within range of an *armoury*, sell all upgrades.
–	**buy <item>**	If within range of an *armoury* and sufficiently wealthy, buy the specified item.
–	**class <class>**	Given sufficient kills, evolve to the specified class.
–	**build <structure>**	As a builder class, build the specified structure.

<item> – *blaster, rifle, ckit, ackit, shotgun, lgun, prifle, mdriver, flamer, chaingun, lcannon, psaw, gren, medkit, jetpack*

<class> – *builder, builderupg, level0, level1, level1upg, level2, level2upg, level3, level3upg, level4*

<structure> – *eggpod, barricade, booster, acid_tube, hive, trapper, overmind, hovel, telenode, medistat, mgturret, tesla, dcc, arm, reactor, repeater*

3.2 Particle System

Files matching the pattern scripts/*.particle are loaded as particle system description files. Each .particle file can contain an arbitrary number of discrete particle systems, much like a .shader file can house many shaders. A particle system is declared by a name followed by curly braces within which the functionality of the particle system is defined. For example:

```
aShinyNewParticleSystem { }
```

Inside the particle system declaration are placed up to four particle ejectors. Ejectors are identified by the keyword ejector and curly braces:

```
aShinyNewParticleSystem
{
  ejector { }
  ejector { }
  thirdPersonOnly
}
```

The thirdPersonOnly keyword may be used to specify that the particle system is not visible from the first person if it relates to that client. The role of the particle ejector is to create some number of new particles at a defined rate. These attributes are controlled by the following parameters:

- *count* <*number*>|*infinite* - the number of particles this ejector will spawn.
- *delay* <*msec*> - the delay in msec before the ejector starts spawning.
- *period* <*initial*> <*final*> <*variance*> - the period between particle ejections.

It is perfectly acceptable to have an initial period of zero. In this case the number of particles specified by the count keyword will be ejected at once. It is not permissible to have count infinite and a period of zero for obvious reasons.

At ejection time each ejector creates up to four new particles based on templates. These are specified in the ejector section using the particle keyword:

```
aShinyNewParticleSystem
{
  ejector
  {
    particle { }
    particle { }
    count 50
    delay 0
    period 0 - 0
  }
}
```

Each particle template has a number of attributes:

- *shader <fps>|sync <shader1> <shader2> ... <shaderN>* - this specifies the shaders to use for the particle. The frame rate can be set to a static rate or the *sync* parameter can be used in which case the frame rate will be synchronised to the lifetime of the particle such that the first frame is displayed on birth and the last frame is displayed immediately before death.
- model <model1> <model2> ... <modelN> - use one of the specified models as the particle. This cannot be used in conjunction with the shader keyword.
- modelAnmation <firstFrame> <numFrames> <loopFrames> <fps>|sync - animation parameters to use when model particles are employed.
- *displacement <x> <y> <z> <variance>* - a static displacement about the attachment point. The *variance* parameter specifies a random displacement in all axes.
- *normalDisplacement <displacement>* - for particle systems that have their normal set (impact particle systems for example) this specifies the magnitude of a displacement along the normal.
- *velocityType static|static_transform|tag|cent|normal* - this specifies how the particle will compute its initial velocity. *static* means it is specified statically in the .particle file, *static_transform means the same, except that it is transformed by the orientation matrix of what it is attached to, tag* means the velocity is in the direction of the tag it is attached to, *cent* means the velocity is in the direction of the cent it is attached to and *normal* means the velocity is in the direction of the particle system normal.
- *velocityDir linear|point* - this specifies whether the initial velocity is computed as a simple direction or as the direction towards a secondary point (defined by *velocityPoint* or dynamically through *velocityType cent*).
- *velocity <x> <y> <z> <variance>* - for when *velocityType static* is present this specifies the direction. The *variance* here is specified in degrees e.g. "~5" - up to 5 degrees deviation.
- *velocityMagnitude <magnitude>* - the magnitude of the velocity.
- *velocityPoint <x> <y> <z> <variance>* - for when *velocityType static* and *velocityDir point* are present this specifies the point to move towards.
- *parentVelocityFraction <fraction>* - for when the particle system is attached to a cent this specifies the fraction of the cent's velocity that is added to the particle's velocity.
- *accelerationType static|static_transform|tag|cent|normal* - this specifies how the particle will compute its acceleration. *static* means it is specified statically in the .particle file, *static_transform means the same, except that it is transformed by the orientation matrix of what it is attached to, tag* means the acceleration is in the direction of the tag it is attached to, *cent* means the acceleration is in the direction of the cent it is attached to and *normal* means the acceleration is in the direction of the particle system normal.
- *accelerationDir linear|point* - this specifies whether the acceleration is computed as a simple direction or as the direction towards a secondary point (defined by *accelerationPoint* or dynamically through *accelerationType cent*).
- *acceleration <x> <y> <z> <variance>* - for when *accelerationType* static is present this specifies the direction. The *variance* here is specified in degrees e.g. "~5" - up to 5 degrees deviation.
- *accelerationMagnitude <magnitude>* - the magnitude of the acceleration.

- *accelerationPoint* <*x*> <*y*> <*z*> <*variance*> - for when *accelerationType static* and *accelerationDir point* are present this specifies the point to move towards.
- *bounce* <*fraction*>|*cull* - the fraction of velocity that is reflected when a particle collides. If this is set to 0.0 the particle won't collide. When *cull* is used particles are culled as soon as they collide with objects.
- bounceMark <count> <radius> <shader> - make a mark at each bounce point for up to <count> bounces.
- bounceSound <count> <sound> - make a sound at each bounce point for up to <count> bounces.
- dynamicLight <delayRadius> <initialRadius> <finalRadius> { <r> <g> <b> } - attach a dynamic light to this particle.
- color <delay> { <ir> <ig> <ib> } { <fr> <fg> <fb> } - color the particle where <i.> refers to the initial color component and <f.> refers to the final color component.
- *overdrawProtection* - cull particles that occupy a large amount of screen space.
- *realLight* - light particles using the lightgrid instead of fullbright.
- *cullOnStartSolid* - cull particles that are spawned inside brushes.
- *radius* <*delay*> <*initial*> <*final*> - the radius of the particle throughout its lifetime. The *delay* parameter specifies the time in msec before radius scaling begins. The *initial* and *final* parameters specify the radii of the particle in quake units.
- *alpha* <*delay*> <*initial*> <*final*> - the alpha of the particle throughout its lifetime. The *delay* parameter specifies the time in msec before alpha scaling begins. The *initial* and *final* parameters specify the alpha of the particle where 1.0 is totally opaque and 0.0 is totally transparent.
- *rotation* <*delay*> <*initial*> <*final*> - the rotation of the particle throughout its lifetime. The *delay* parameter specifies the time in msec before the rotation begins. The *initial* and *final* parameters specify the rotation of the particle in degrees.
- *lifeTime* <*time*> - the lifetime of the particle.
- *childSystem* <*particle system*> - specifies a particle system to attach to this particle.
- childTrailSystem <trail system> - specifies a trail system to attach to this particle.
- *onDeathSystem* <*particle system*> - specifies a particle system to spawn at the point where this particle died.

Except for vector components, *shader fps ...* and *period* <*initial* <*final*> <*variance*>, every value can be specified with a random variance. The syntax for this is as follows:

```
[value][~variance[%]]
```

So the following forms are possible, where random is a random number between 0.0 and 1.0 inclusive:

```
5.0      // 5.0
5.0~8.0  // 5.0 + ( random * 8.0 )
5.0~200% // 5.0 + ( random * 5.0 * 200% )
~7.0     // random * 7.0
```

This allows for relatively flexible randomisation of most of the particle's parameters. For parameters taking an initial and final value, specifying the final value as '-' will result in a final value the same as the initial value.

For the purposes of map based particle systems using *misc_particle_system* it is safe to ignore *velocityType* and *accelerationType tag|cent|normal*, *normalDisplacement* and *parentVelocityFraction* altogether.

Of course, it is not necessary to specify every parameter documented here for every particle system. If a parameter is not included it will usually default to zero. C/C++ style comments can be used throughout. There are an enormous number of possible combinations of particle systems parameters and as such it is impractical to test them all. For this reason it is possible that certain permutations do not behave as expected or wrongly. In this case you may have discovered a bug - let us know. Having said this when you're having problems with a particle system make sure you scroll up the console and check that it compiled OK, I've written the parser to be very intolerant of error.

Here is an example particle system:

```
aShinyNewParticleSystem
{
  ejector
  {
    particle
    {
      shader sync shader1 shader2

      velocityType static
      velocityDir linear
      velocityMagnitude 200
      velocity 0 0 1 ~30
      accelerationType static
      accelerationDir linear
      accelerationMagnitude 50
      acceleration 0 0 1 ~0
      radius 0 10.0 50.0
      alpha 0 1.0 1.0
      rotation 0 ~360 -
      bounce 0.4
      lifeTime 1500
    }
    count 50
    delay 0
    period 0 - 0
  }
}
```

3.3 Trail System

Files matching the pattern scripts/*.trail are loaded as trail system description files. Each .trail file can contain an arbitrary number of discrete trail systems, much like a .shader file can house many shaders. A trail system is declared by a name followed by curly braces within which the functionality of the trail system is defined. For example:

```
aShinyNewTrailSystem { }
```

Inside the particle system declaration are placed up to four trail beams. Beams are identified by the keyword beam and curly braces:

```
aShinyNewTrailSystem
{
  beam { }
  beam { }
  thirdPersonOnly
}
```

The thirdPersonOnly keyword may be used to specify that the trail system is not visible from the first person if it relates to that client. A trail beam describes the appearance of one element of the trail system:

- *shader <shader> - the shader to use to texture this beam.*
- segments <number> - the number of quads that make up the beam.
- width <frontWidth> <backWidth> - the width of the beam at the front and back.
- alpha <frontAlpha> <backAlpha> - the alpha of the beam at the front and back.
- color { <fr> <fg> <fb> } {
 <bg> <bb> } - the color of the beam at the front and back.

- segmentTime <time> - how long a single segment lasts when the trail is only attached at one end.
- fadeOutTime <time> - how long this beam takes to fade away.
- textureType [stretch <frontTC> <backTC>]|[repeat [front|back] <repeatLength>] - how to texture the beam. stretch causes the texture to be stretched from the front to the back using the specified texture coordinates. repeat causes the texture to be repeated over a specified length either from the front or the back.
- model <model1> <model2> ... <modelN> - use one of the specified models as the particle. This cannot be used in conjunction with the shader keyword.
- modelAnmation <firstFrame> <numFrames> <loopFrames> <fps>|sync - animation parameters to use when model particles are employed.
- *realLight* - light particles using the lightgrid instead of fullbright.
- jitter <magnitude> <period> - this specifies a random jitter of the position of each beam node by magnitude every period.
- jitterAttachments - if this is specified the end points of the beam are jittered as well as the intervening nodes.

3.4 Map Rotation System

The file *maprotation.cfg* is used to describe up to 16 map rotations which may be used by a Tremulous server. In its most simple form, the syntax is as follows:

```
mapRotation1
{
  map1
  map2
  map3
}
mapRotation2
{
  map6
  map3
  map9
}
```

This specifies two rotations, each consisting of three maps. The contents of the cvar *g_initialMapRotation* specifies the map rotation to start after the map the server was started with has finished. It is possible to specify a list of server commands to be run after a map has finished:

```
mapRotation3
{
  map1
  {
    set sv_hostname "Just finished map1!"
    set g_teamForceBalance 0
  }

  map2
  {
    set g_teamForceBalance 1
  }
  map3
}
```

Primitive logic is also available:

```
mapRotation4
{
  map1
  goto map3
  map2
  if numClients > 8
    mapRotation3
  map3
  if lastWin aliens
    mapRotation2

  if random
    mapRotation1
}
mapRotation5
{
  map1
  if lastWin humans
    map4
  map2
  map3
  goto map1
  map4
  map5
}
```

The **goto** keyword is used to unconditionally branch to either another map *in the current rotation* or another map rotation entirely. The **if** keyword is used in conjunction with a condition to decide whether or not to branch to the specified map or rotation (as with the **goto** keyword). The condition itself can be one of **numClients <op> <number>**, **lastWin <team>** or **random**, where **<op>** is **<**, **>** or **=** and **<team>** is **aliens** or **humans**. The **random** condition simply chooses whether or not to execute the change randomly, with each outcome equally likely.

4 Credits

Tim *'Timbo'* Angus – Programming and Direction

Nick *'jex'* Jansens – Mapping, texturing and 2D artwork

Robin *'OverFlow'* Marshall – Modelling, animation and mapping

Jan *'Stannum'* van der Weg – Texturing and mapping

Mike *'Veda'* Mcinnerney – Modelling, animation and texturing

Gordon *'Godmil'* Miller – Mapping

'Who-[Soup]' – Mapping

Tristan *'jhrx'* Blease – Mapping

Paul *'MoP'* Greveson – Modelling and texturing

Chris *'Dolby'* McCarthy – Sound

Special thanks

Asa *'Norfenstein'* Kravets – Manual content, QA, design and balance suggestions

'Crylar' – Concept art

Yves *'evillair'* Allaire – Textures

Randy *'ydnar'* Reddig – Textures

Richard *'R1CH'* Stanway – Server hosting

Stéphane *'MEGASTeP'* Peter – Early test server hosting

Sourceforge and TARDIS – Web hosting

icculus.org – Subversion hosting

The contributors to icculus.org/quake3/ – Various

Arsonide, Bajoran, Bt, Chamooze, Crylar, Cybernetsam, dzjepp, ectox, Edo, evil poop, Excalibur, FroggyQuim, Idle Wild, juice, Kai, kingping, Lava Croft, MajorPain, MiDiaN, Molog, Mutemode, Norfenstein, Orc, R1CH, Ratti, Ravyn, Salteh, Sandy, SharkDog, slux, Suddien, Supa, Survivor, Swie, sysrq, TerrorEast, Tyler, Vitae, Woo – Beta testers

Also thanks

babyomen, Carc, djbob, Grim, Grytviken, Gumby, heimdall, Hellbringer, Hentai, Mighty_Pea, Psylo, Reaper-1, RR2D02, Saig, Smack, T-bone, The GtkRadiant people, The inhabitants of Quake3World, ThePyro, TTimo, ValouR